

Malware analysis

S10

Team Alba

Indice generale

01

Analisi Statica
Basica

02

Analisi Dinamica
Basica

03

Introduzione ad
Assembly

04

Assembly:
Tecniche di
Ingegneria inversa

05

Analisi Statica e
Dinamica:
approccio pratico



GIORNO 1

L1

Analisi Statica Basica

Traccia giorno 1

Con riferimento al file eseguibile contenuto nella cartella «Esercizio_Pratico_U3_W2_L1» presente sul Desktop della vostra macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti:

01

Indicare le librerie importate dal malware, fornendo una descrizione per ognuna di esse

02

Indicare le sezioni di cui si compone il malware, fornendo una descrizione per ognuna di essa

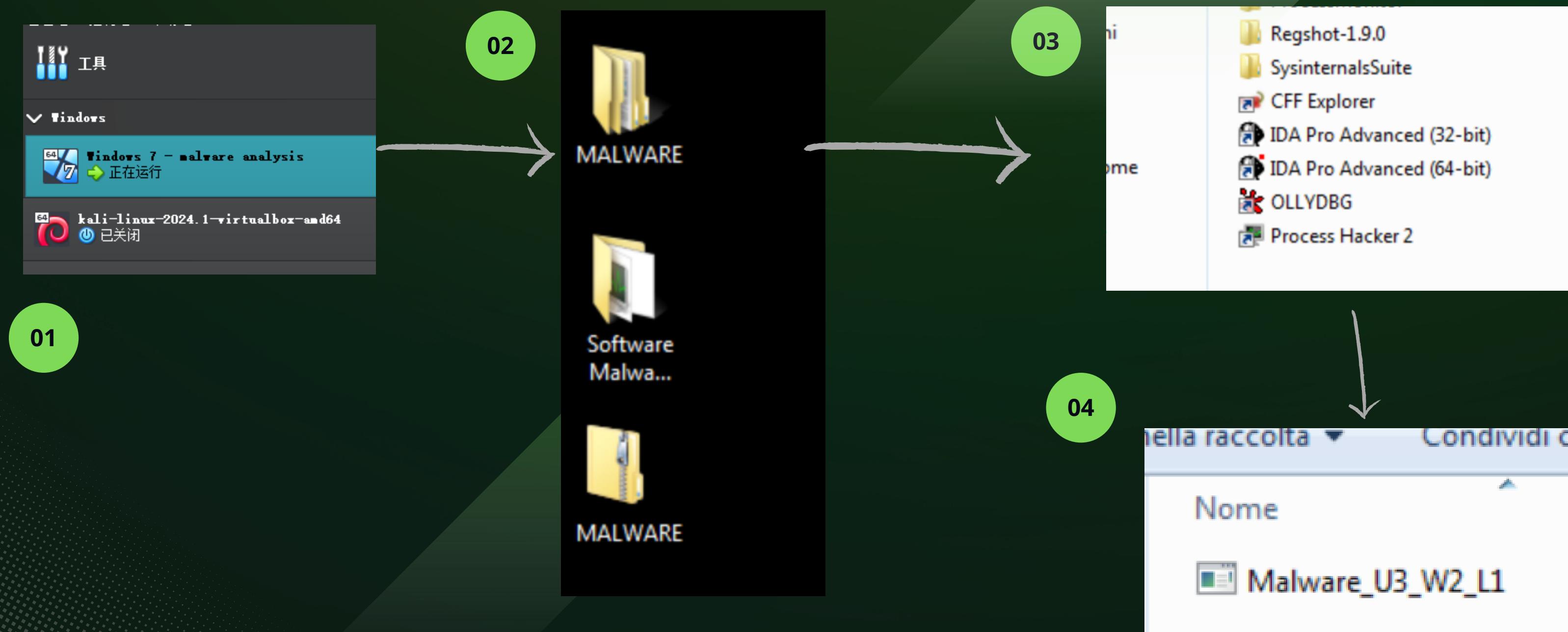
03

Aggiungere una considerazione finale sul malware in analisi in base alle informazioni raccolte

Per utilizzare Cff per analizzare il malware, è necessario seguire questi passaggi:

1. Scaricare e installare Windows 7.
2. Aprire la cartella "soft malware".
3. Avviare Cff Explorer.
4. Importare il file "Malware_U3_W2_L1" in Cff Explorer.

Questi passaggi permetteranno di analizzare il malware utilizzando Cff Explorer su Windows



CFF Explorer

CFF Explorer è uno strumento avanzato per l'analisi e la modifica di file eseguibili su Windows, parte della suite di strumenti chiamata Explorer Suite, sviluppata da NTCore. È particolarmente utile per programmati, analisti di malware e ricercatori di sicurezza informatica. Ecco alcune delle sue funzionalità principali:

1. Visualizzazione della struttura dei file PE: Permette di esplorare e modificare le intestazioni e le sezioni dei file Portable Executable (PE), come .exe e .dll.
2. Modifica degli import e degli export: Consente di visualizzare e modificare le tabelle degli import e degli export, essenziali per comprendere le dipendenze di un programma.
3. Risorse del file: Permette di visualizzare e modificare le risorse incorporate nel file, come icone, immagini, stringhe di testo e altri dati.
4. Editor HEX: Include un editor esadecimale per la modifica diretta dei dati binari del file.
5. Disassemblatore: Offre funzionalità di disassemblaggio per analizzare il codice macchina del file eseguibile.

CFF Explorer è utilizzato frequentemente nell'analisi di malware perché consente di esaminare la struttura interna dei file eseguibili sospetti, identificare potenziali comportamenti dannosi e apportare modifiche per ulteriori analisi o mitigazioni.

LIBRERIE IMPORTATE

elenca tutte le funzioni e le librerie esterne che un file eseguibile o una libreria dinamica (.exe o .dll) necessita per funzionare correttamente.

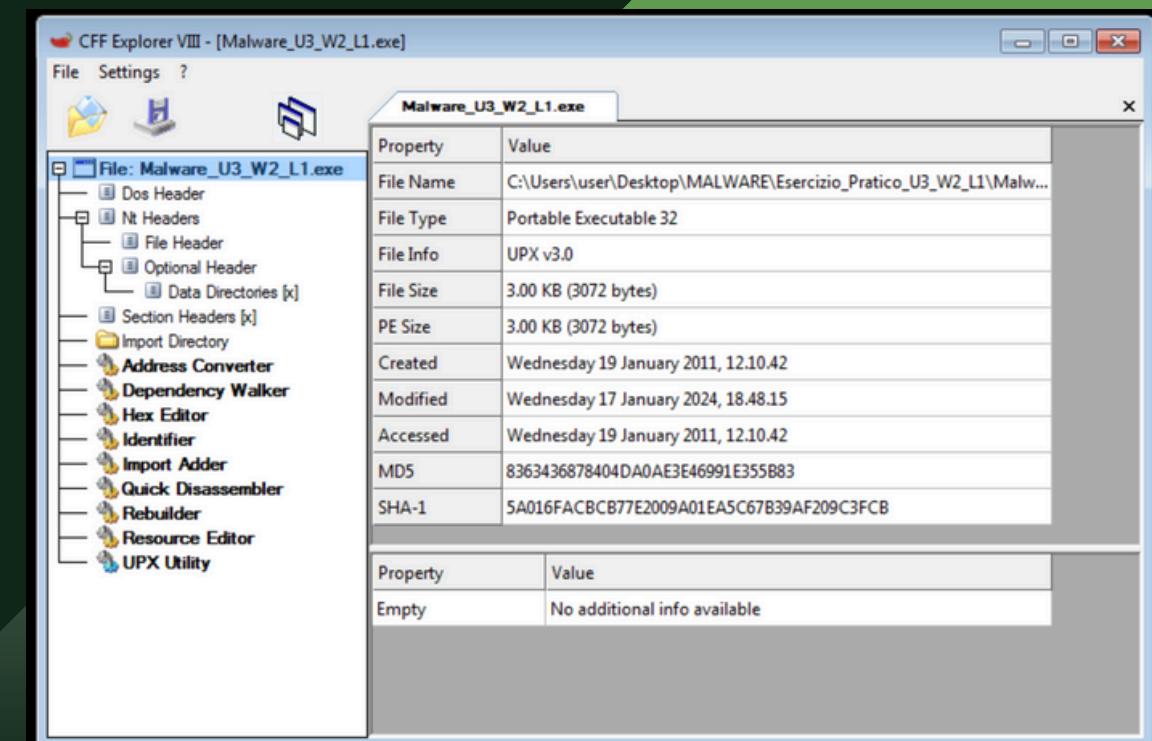
Utilizzando CFF Explorer, vediamo dalla sezione import directory che il malware U3_W2_L1 importa 4 librerie:

1. Kernel32.dll, che include le funzioni core del sistema operativo

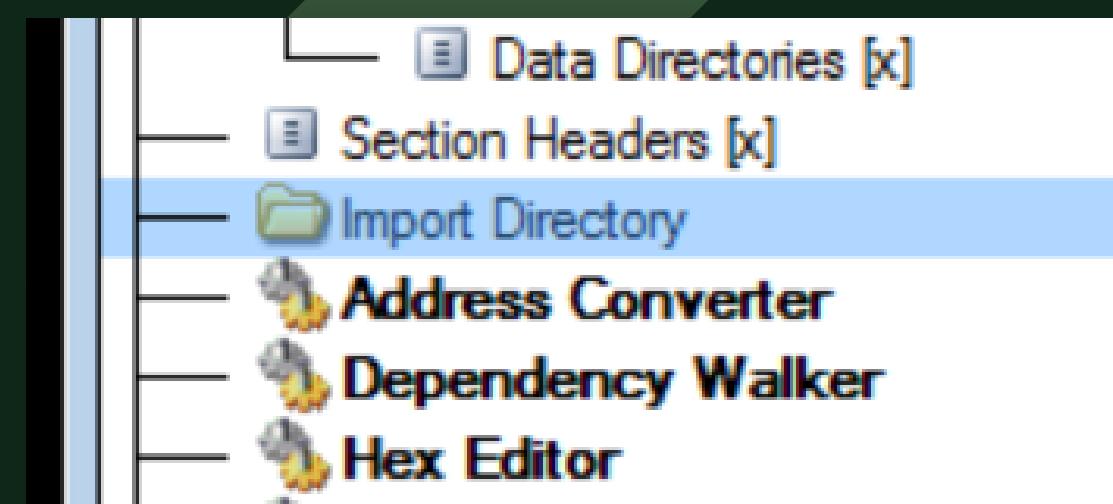
2. Advapi32.dll, che include le funzione per interagire con registri e servizi Windows

3. MSVCRT.dll, libreria scritta in C per la manipolazione scritte o allocazione memoria

4. Wininet.dll, include le funzione per implementare i servizi di rete come ftp, ntp, http



Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.DLL	6	00000000	00000000	00000000	00006098	00006064
ADVAPI32.dll	1	00000000	00000000	00000000	000060A5	00006080
MSVCRT.dll	1	00000000	00000000	00000000	000060B2	00006088
WININET.dll	1	00000000	00000000	00000000	000060BD	00006090



SECTION HEADER

testazione delle sezioni (section header) di un file eseguibile (PE, Portable Executable) su Windows è una parte cruciale che descrive le caratteristiche delle diverse sezioni del file. Ogni sezione può contenere codice, dati, risorse o altre informazioni necessarie per l'esecuzione del programma.

Da CFF Explorer, dalla sezione «section header» vediamo che l'eseguibile si compone di 3 sezioni. Purtroppo sembra che il malware abbia nascosto il vero nome delle sezioni e quindi non siamo in grado di capire che tipo di sezioni sono.

Malware_U3_W2_L1.exe									
Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
UPX0	00004000	00001000	00000000	00000400	00000000	00000000	0000	0000	E0000080
UPX1	00001000	00005000	00000600	00000400	00000000	00000000	0000	0000	E0000040
UPX2	00001000	00006000	00000200	00000A00	00000000	00000000	0000	0000	C0000040

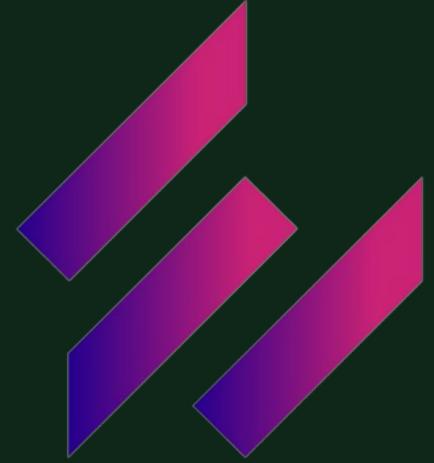
Considerazione finale

Si tratta di un malware avanzato che non ci consente di recuperare molte informazioni sul suo comportamento con l'analisi statica basica.

Ciò è supportato dal fatto che tra le funzioni importate troviamo «LoadLibrary e GetProcAddress», che ci fanno pensare ad un malware che importa le librerie a tempo di esecuzione (runtime) nascondendo di fatto le informazioni circa le librerie importate a monte.

Module Name	Imports	OFTs	TimeStamp
00000A98	N/A	00000A00	00000A04
szAnsi	(nFunctions)	Dword	Dword
KERNEL32.DLL	6	00000000	00000000
ADVAPI32.dll	1	00000000	00000000
MSVCRT.dll	1	00000000	00000000
WININET.dll	1	00000000	00000000

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
N/A	000060C8	0000	LoadLibraryA
N/A	000060D6	0000	GetProcAddress
N/A	000060E6	0000	VirtualProtect
N/A	000060F6	0000	VirtualAlloc
N/A	00006104	0000	VirtualFree



GIORNO 2

L2

Analisi Dinamica Basica

Traccia giorno 2

Configurare la macchina virtuale per l'analisi dinamica (il malware sarà effettivamente eseguito).

Con riferimento al file eseguibile contenuto nella cartella «Esercizio_Pratico_U3_W2_L2» presente sul desktop della vostra macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti:

- 01 Identificare eventuali azioni del malware sul file system utilizzando Process Monitor (procmon)
- 02 Identificare eventuali azioni del malware su processi e thread utilizzando Process Monitor
- 03 Modifiche del registro dopo il malware (le differenze)
- 04 Provare a profilare il malware in base alla correlazione tra «operation» e Path.

Configurazione macchina virtuale

In “rete” disabilitiamo tutte le interfacce di rete:

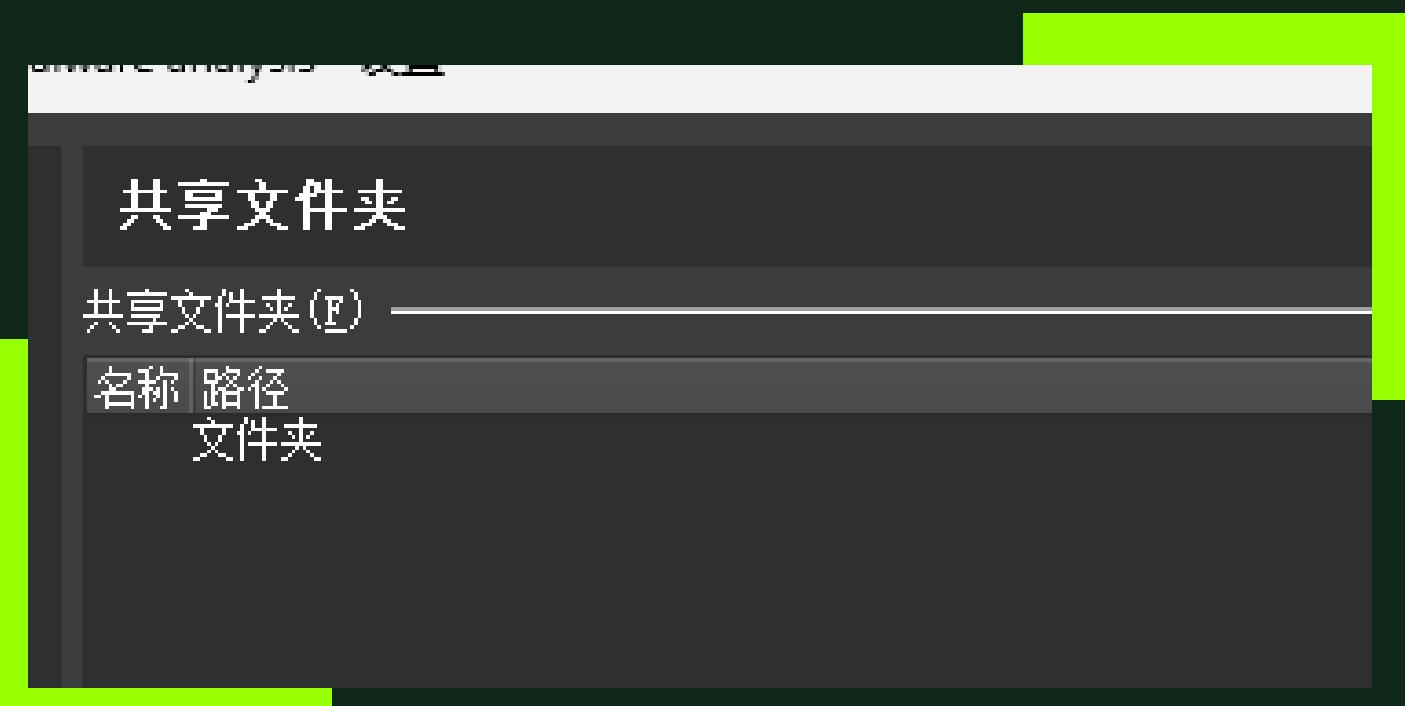
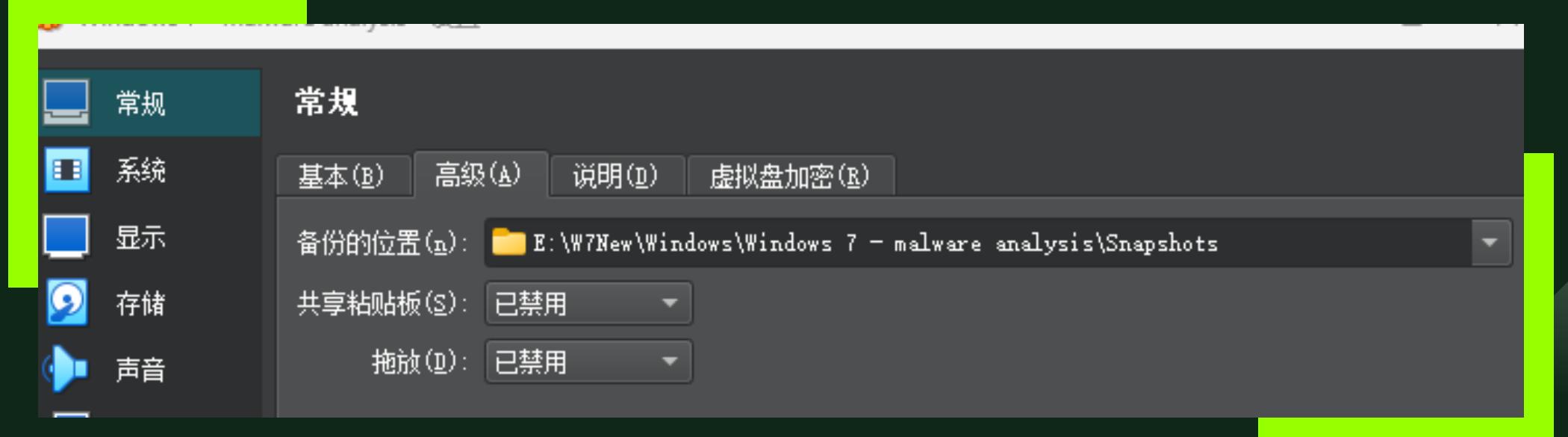


In “Sistema” disattiviamo il Floppy per poi disabilitare il controller USB in “Porte” > “USB”:

The image contains two side-by-side screenshots of a virtual machine configuration interface. The left screenshot shows the 'System' tab with the 'Boot Order' section. The 'Floppy' option is checked with a red circle and question mark icon, indicating it is disabled. The right screenshot shows the 'Hardware Analysis' window with the 'USB 设备' (USB Devices) tab selected. It displays options for enabling the USB controller and selecting between three different controller types: 'USB 1.1 (OHCI) 控制器', 'USB 2.0 (OHCI + EHCI) 控制器', and 'USB 3.0 (xHCI) 控制器'. The 'USB 1.1 (OHCI) 控制器' radio button is selected. Both screenshots have a large green rectangular box highlighting the main configuration area.

Configurazione macchina virtuale

In “Generale” > “Avanzate” disabilitare “appunti condivisi” e “trascina e rilascia”:



Configurazione macchina virtuale

In



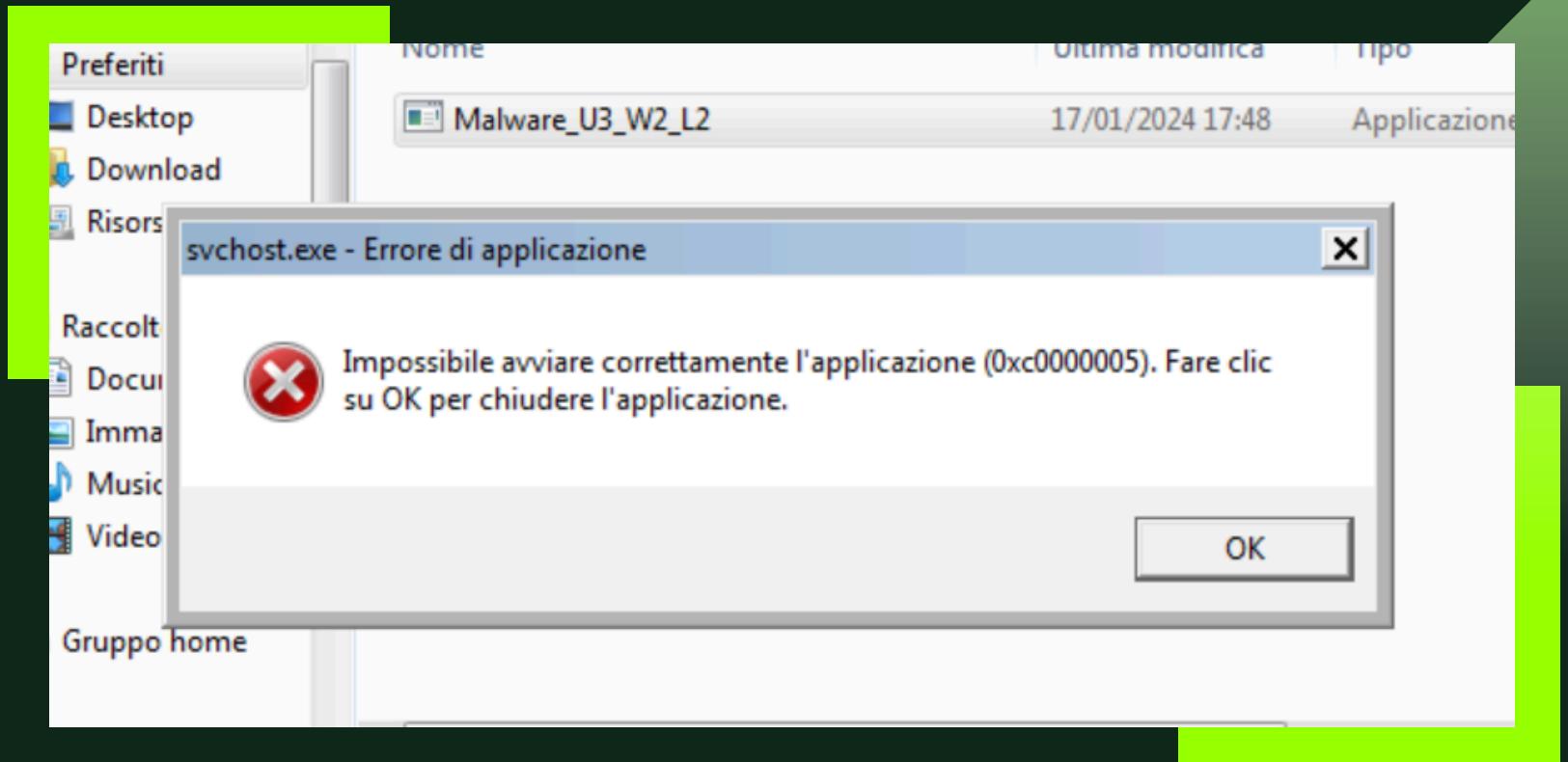
> “Istantanee” > “Crea”, creiamo un’istantanea della VM.

Questa operazione serve per avere un backup dello stato attuale della macchina in caso dopo l’esecuzione del malware questa venga compromessa.

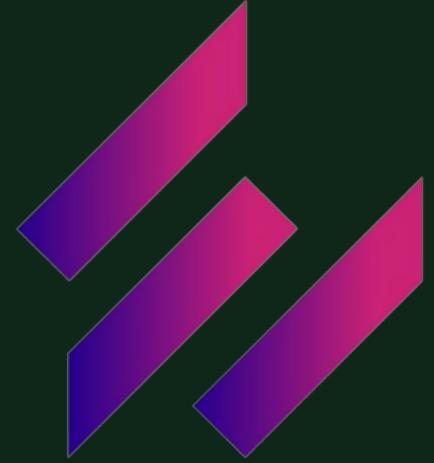


Esecuzione del malware

Provando ad eseguire il Malware, il sistema ci restituisce il seguente errore:



Dopo svariate ricerche abbiamo appurato che il Malware è eseguibile su Windows XP e non su Windows 7. A questo punto abbiamo cercato un modo per copiare i file contenenti il malware da Windows 7 a Windows XP e lo abbiamo fatto creando un disco fisso VDI vuoto su VirtualBox. Abbiamo montato il disco vuoto su Windows 7 e abbiamo copiato su di esso i dati di nostro interesse, dopodiché abbiamo montato lo stesso disco (che a questo punto contiene i dati) su WindowsXP.



GIORNO 3

L3

La memoria ed il linguaggio Assembly

Traccia giorno 3

Dato il codice in Assembly per la CPU x86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice.

01 0x00001141 <+8>: mov EAX,0x20
0x00001148 <+15>: mov EDX,0x38
0x00001155 <+28>: add EAX,EDX
0x00001157 <+30>: mov EBP, EAX
0x0000115a <+33>: cmp EBP,0xa
0x0000115e <+37>: jge 0x1176 <main+61>
0x0000116a <+49>: mov eax,0x0
0x0000116f <+54>: call 0x1030 <printf@plt>

Linguaggio Assembly

La base del linguaggio Assembly sono le istruzioni. Nel linguaggio Assembly le istruzioni sono costituite da due parti:

- Un codice mnemonico, ovvero una parola che identifica l'istruzione da eseguire
- Uno o più operandi (per alcuni codici mnemonici, non è necessario l'operando come vedremo a breve), che identificano le variabili o la memoria oggetto dell'istruzione.

Un linguaggio assembly (detto anche linguaggio assemblativo o linguaggio assemblatore o semplicemente assembly) è un linguaggio di programmazione molto simile ai linguaggi macchina.

Si differenzia da questi ultimi principalmente per l'utilizzo di identificatori mnemonici, valori simbolici e altre caratteristiche che lo rendono più agevole da scrivere e leggere per gli esseri umani.

Erroneamente viene spesso chiamato assembler, ma quest'ultimo termine identifica solo l'applicativo che converte i programmi scritti in assembly nell'equivalente in linguaggio macchina.

Conversione esadecimale -> decimale

Il sistema numerico esadecimale utilizza le cifre da 0 a 9 e le lettere da A a F (10-15). Per convertire un numero da base esadecimale a base decimale, è sufficiente moltiplicare la cifra o il carattere per 16 elevato alla posizione in cui si trova la cifra o il carattere proseguendo da destra verso sinistra.

Ecco un esempio pratico per chiarire il concetto:

3B ---->numero in base esadecimale

$$B = B \times 16^0 = 11 \times 16^0 = 11$$

$$3 = 3 \times 16^1 = 48$$

$11 + 48 = 59$ ----> numero corrispondente in base decimale

Conversione decimale -> esadecimale

Il sistema numerico decimale utilizza le cifre da 0 a 9. Per convertire un numero da base decimale a base esadecimale, è sufficiente dividere il numero per 16, se il quoziente è diverso da zero va diviso nuovamente per 16 finché non sarà uguale a 0. A questo punto bisogna prendere il resto di ogni divisione in ordine opposto di come li abbiamo ottenuti.

Ecco un esempio pratico per chiarire il concetto:

59 ----->numero in base decimale

$59 : 16 = 3$ con resto 11

$3 : 16 = 0$ con resto 3

In riga scriviamo tutti i resti ottenuti, dall'ultimo al primo:

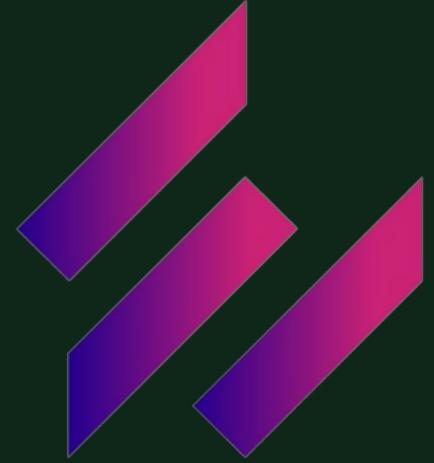
3 11

Sappiamo che in base esadecimale $B = 11$ e quindi il risultato finale sarà

3B -----> numero in base esadecimale

Descrizione codice

- 1.0x00001141 <+8>: mov EAX, 0x20 #Sposta il valore 0x20 (32 in decimale) nel registro EAX. Quindi EAX = 32
- 2.0x00001148 <+15>: mov EDX, 0x38 #Sposta il valore 0x38 (56 in decimale) nel registro EDX. Quindi EDX = 56
- 3.0x00001155 <+28>: add EAX, EDX #Aggiunge il valore in EDX a EAX. EAX = EAX + EDX = 32 + 56 = 88 (0x58)
- 4.0x00001157 <+30>: mov EBP, EAX #Sposta il contenuto di EAX in EBP. Quindi 88, che è il nuovo valore di EAX viene spostato in EBP. EBP = 88
- 5.0x0000115a <+33>: cmp EBP, 0xA #Confronta il valore in EBP con 0xA (10). EBP>0xA ---> 88 > 10
- 6.0x0000115e <+37>: jge 0x1176 <main+61> # jge, jump if greater or qual, Salta all'indirizzo 0x1176 se la relazione tra gli operatori valutati risulta essere maggiore o uguale. Questo prende il nome di salto condizionato. EBP > 0xA e quindi il salto viene effettuato
- 7.0x0000116a <+49>: mov EAX, 0x0 #Sposta il valore 0 in EAX. Quindi EAX = 0
- 8.0x0000116f <+54>: call 0x1030 <printf@plt> #Chiama la funzione printf allocata all'indirizzo 0x1030, e in questo caso non verrà eseguita perché è stato effettuato il salto direttamente all'indirizzo 0x1176



GIORNO 4

ЛЧ

Linguaggio Assembly parte 2

Traccia giorno 4

Traccia: La figura seguente mostra un estratto del codice di un malware. Identificare i costrutti noti Esercizio Linguaggio Assembly visti durante la lezione teorica.

```
* .text:00401000          push    ebp
* .text:00401001          mov     ebp, esp
* .text:00401003          push    ecx
* .text:00401004          push    0           ; dwReserved
* .text:00401006          push    0           ; lpdwFlags
* .text:00401008          call    ds:InternetGetConnectedState
* .text:0040100E          mov     [ebp+var_4], eax
* .text:00401011          cmp     [ebp+var_4], 0
* .text:00401015          jz      short loc_40102B
* .text:00401017          push    offset aSuccessInterne ; "Success: Internet Connection\n"
* .text:0040101C          call    sub_40105F
* .text:00401021          add    esp, 4
* .text:00401024          mov     eax, 1
* .text:00401029          jmp    short loc_40103A
.text:0040102B ; -----
.text:0040102B ; -----
```

Identificare i costrutti

1. Identificare i costrutti noti (e s. while, for, if, switch, ecc.)

```
1.    00401000      push  ebp  
2.    00401001      mov   ebp,esp  
  
3.    00401003      push  ecx  
4.    00401004      push  0 ; dwReserved  
5.    00401006      push  0 ; lpdwFlags  
6.    00401008      call   ds:InternetGetConnectedState  
  
7.    0040100E      mov   [ebp+var_4], eax  
  
8.    00401011      cmp   [ebp+var_4],0  
9.    00401015      jz    short loc_40102B  
10.   00401017      push  offset aSuccessInterne ; "Succ....\n"  
11.   0040101C      call   sub_40105F  
12.   00401021      add   esp,4  
13.   00401024      mov   eax,1  
  
14.   00401029      jmp   short loc_40103A  
15.   0040102B  
16.   0040102B
```

Costrutto if



Ipotizzare la funzionalità

2. Ipotizzare la funzionalità – esecuzione ad alto livello

Questa porzione di codice assembly è progettata per verificare lo stato della connessione a Internet su un sistema Windows. Utilizza la funzione InternetGetConnectedState della libreria WinINet per determinare se il sistema è attualmente connesso a Internet e agisce di conseguenza.

Il malware chiama la funzione internetgetconnectedstate e ne controlla con un «if» il valore di ritorno. Se il valore di ritorno (return) della funzione è diverso da 0, allora vuol dire che c'è una connessione attiva.

Pseudocodice C:

```
state = internetgetconnectedstate (par1,0,0);
If (state !=0) printf ("Active connection");
Else return 0;
```



Bonus

3. BONUS: studiare e spiegare ogni singola riga di codice

01

Impostazione del Frame dello Stack

push ebp

mov ebp, esp

- Salva il valore corrente del puntatore di base (ebp) sullo stack.
- Imposta il puntatore di base (ebp) all'attuale puntatore dello stack (esp), creando un nuovo frame dello stack.

02

Salvataggio del Registro ecx

push ecx

- Salva il valore del registro ecx sullo stack per preservarlo, poiché sarà utilizzato nel corso della funzione.

03

Preparazione dei Parametri per InternetGetConnectedState

push 0 ; dwReserved

push 0 ; lpdwFlags

Imposta i parametri richiesti dalla funzione InternetGetConnectedState:

- lpdwFlags è un puntatore a una variabile che riceve la descrizione della connessione, ma qui non viene utilizzato (impostato a 0).
- dwReserved è riservato e deve essere 0.



Bonus

3. BONUS: studiare e spiegare ogni singola riga di codice

04

La Chiamata alla Funzione InternetGetConnectedState

```
call ds:InternetGetConnectedState mov [ebp+var_4],
```

eax Chiama la funzione InternetGetConnectedState e memorizza il valore di ritorno nel registro eax.

Questo valore indica se il sistema ha una connessione a Internet attiva. Memorizza il risultato in una variabile locale (var_4) nello stack.

05

Verifica dello Stato della Connessione Internet

```
cmp [ebp+var_4], 0
```

```
jz short loc_40102B
```

- Confronta il valore memorizzato in var_4 con 0.
- Se var_4 è 0 (nessuna connessione a Internet), salta all'etichetta loc_40102B.

06

Stampa del Messaggio di Successo

```
push offset aSuccessInterne ; "Success\n"
```

```
call sub_4010fF
```

```
add esp, 4
```

- Se viene rilevata una connessione a Internet, impila l'indirizzo della stringa "Success\n" e chiama la funzione sub_40105F (presumibilmente una funzione che stampa o gestisce la stringa).
- Dopo la chiamata, regola il puntatore dello stack (esp).



Bonus

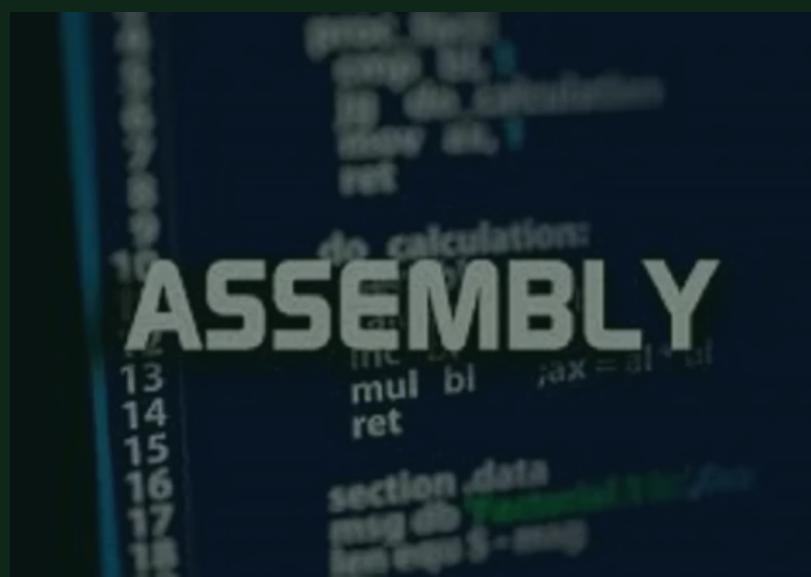
3. BONUS: studiare e spiegare ogni singola riga di codice

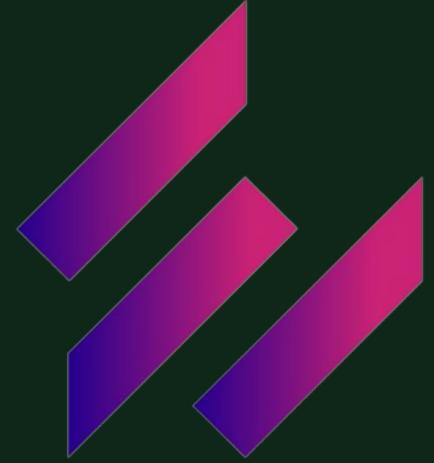
07

Impostazione del Valore di Ritorno e Uscita

```
mov eax, 1  
jmp short loc_40103A
```

- Imposta il valore di ritorno in eax a 1 (indicando successo) e salta all'etichetta loc_40103A per uscire dalla funzione.





S10_L5

Progetto S10_L5

Indice generale

- 
- 01 ANALISI LIBRERIE DEL MALWARE _U3_W2_L5
 - 02 ANALISI SEZIONI DEL MALWARE _U3_W2_L5
 - 03 IDENTIFICAZIONE COSTRUTTI
 - 04 IPOTESI COMPORTAMENTO FUNZIONALITÀ
 - 05 BONUS TABELLA RIGHE DI CODICE ASSEMBLY

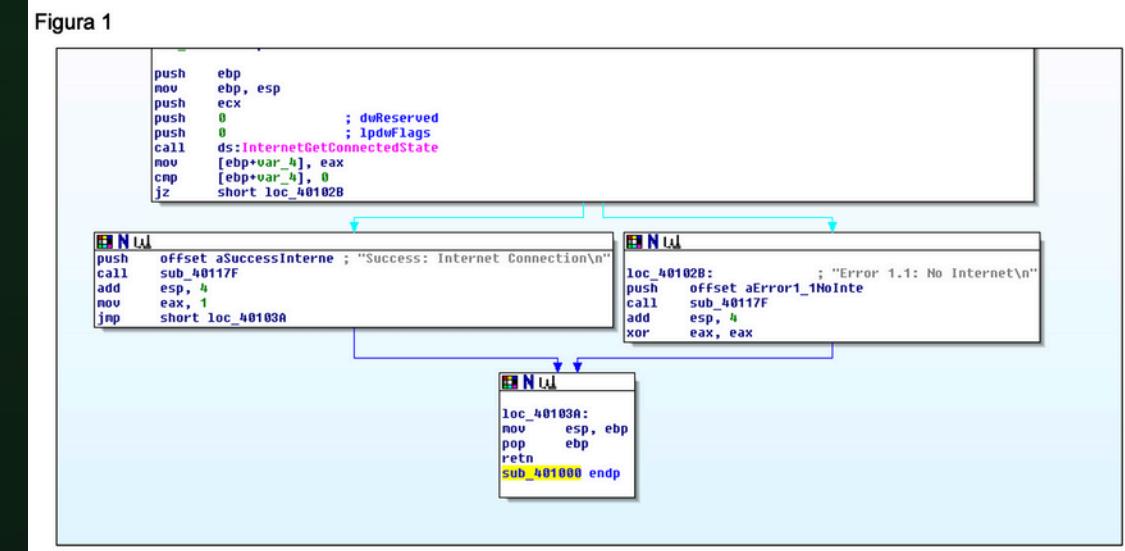
Traccia giorno 5

Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile?
2. Quali sono le sezioni di cui si compone il file eseguibile del malware?

Con riferimento alla figura 1, rispondere ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli)
4. Ipotizzare il comportamento della funzionalità
5. BONUS fare tabella con significato delle singole righe di codice Esercizio Traccia e requisiti altri costrutti) implementata assembly.



1 - Malware_U3_W2_L5

Librerie

Per analizzare le librerie di malware Malware_U3_W2_L5, utilizziamo il tool CFF Explorer. Abbiamo individuato le seguenti librerie chiave:

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

KERNEL32.dll

Questa libreria contiene numerose funzioni essenziali per il funzionamento del sistema operativo e delle applicazioni Windows. Le principali aree di gestione includono:

- **Memoria:** Funzioni come VirtualAlloc, VirtualFree, HeapCreate, e HeapAlloc sono utilizzate per l'allocazione e la liberazione della memoria.
- **Processi e Thread:** Funzioni come CreateProcess, CreateThread, TerminateProcess, e TerminateThread permettono la creazione e gestione di processi e thread.
- **Input/Output:** Funzioni come ReadFile, WriteFile, CreateFile, e CloseHandle gestiscono operazioni di lettura, scrittura e gestione di file e dispositivi.
- **File:** Funzioni come CopyFile, DeleteFile, e MoveFile gestiscono la manipolazione dei file nel file system.
- **Tempi e Date:** Funzioni come GetSystemTime, SetSystemTime, e GetTickCount forniscono e manipolano informazioni temporali.
- **Risorse:** Funzioni come LoadLibrary, GetProcAddress, e FreeLibrary permettono di caricare, ottenere indirizzi di funzioni e liberare librerie dinamiche.

Nel contesto del malware, queste funzioni possono essere utilizzate per vari scopi malevoli, come:

- **Persistenza:** Il malware può utilizzare funzioni di gestione dei processi e dei thread per rimanere attivo nel sistema anche dopo il riavvio.
- **Evasione:** Funzioni di gestione della memoria possono essere sfruttate per nascondere il malware o offuscare il suo codice.
- **Comunicazione:** Operazioni di input/output e gestione dei file possono essere utilizzate per esfiltrare dati o comunicare con server di comando e controllo.

1 - Malware_U3_W2_L5

Librerie

Per analizzare le librerie di malware Malware_U3_W2_L5, utilizziamo il tool Explorer. Abbiamo individuato le seguenti funzioni chiave:

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

WININET.dll

Questa libreria contiene funzioni per l'implementazione di vari protocolli di rete, come HTTP, FTP e NTP. Fornisce un'API che consente alle applicazioni Windows di:

- Effettuare richieste di rete: Funzioni come InternetOpen, InternetConnect, HttpOpenRequest, e HttpSendRequest permettono di inviare richieste HTTP ai server web.
- Gestire operazioni di rete: Funzioni come FtpGetFile, FtpPutFile, e InternetReadFile facilitano il download e l'upload di file tramite FTP e la lettura di dati da internet.

In un contesto di malware, le funzioni di WININET.dll possono essere sfruttate per:

- Download e Upload di Payload: Il malware può utilizzare funzioni di rete per scaricare componenti aggiuntivi o caricare dati rubati.
- Comunicazione con Server C2: Le richieste HTTP possono essere usate per comunicare con server di comando e controllo per ricevere istruzioni o inviare dati.

1 - Malware_U3_W2_L5

KERNEL32

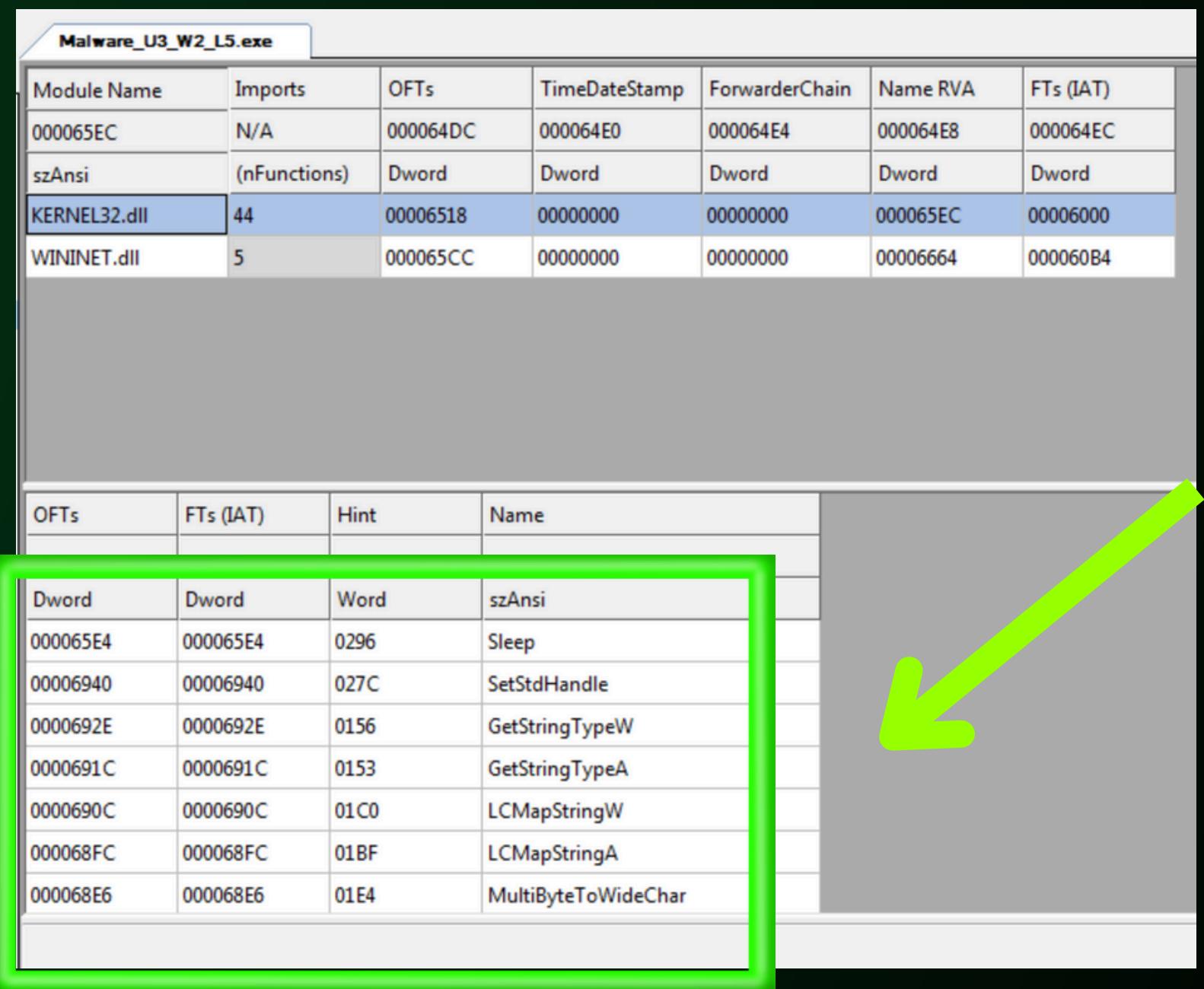
Malware_U3_W2_L5.exe						
Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	000064EC
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

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	LCMMapStringW
000068FC	000068FC	01BF	LCMMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar

La libreria **KERNEL32**, parte integrante del sistema operativo Windows, fornisce una vasta gamma di funzionalità essenziali per la gestione delle risorse di sistema e l'esecuzione dei programmi. Tra le 44 funzioni che importa, alcune delle più significative includono:

- **Sleep**: Questa funzione di sistema sospende temporaneamente l'esecuzione di un thread per un periodo specificato. Durante questo tempo, altri thread possono essere eseguiti, permettendo una gestione più efficiente delle risorse di calcolo. Ad esempio, un thread che non ha bisogno di essere eseguito continuamente può essere messo in pausa, liberando tempo di CPU per altri processi.
- **CloseHandle**: È un'API utilizzata per chiudere un handle, ossia un riferimento o puntatore a un oggetto kernel. Questa funzione è cruciale per la gestione delle risorse, poiché rilascia le risorse allocate a un handle una volta che non sono più necessarie, prevenendo perdite di memoria e migliorando la stabilità del sistema.

- **GetProcAddress**: Questa funzione permette ai programmatori di ottenere un puntatore a una funzione specifica all'interno di una DLL (Dynamic Link Library) caricata in memoria. Viene utilizzata per chiamare funzioni esportate da una DLL in modo dinamico durante l'esecuzione del programma, anziché durante la fase di compilazione. Questo è particolarmente utile per plugin, estensioni o per caricare moduli opzionali senza ricompilare l'intera applicazione.
- **VirtualAlloc**: Utilizzata principalmente per riservare o allocare memoria virtuale per un processo. Questa funzione consente di riservare una porzione di memoria virtuale senza dover allocare immediatamente la memoria fisica corrispondente. È fondamentale per la gestione efficiente della memoria, poiché permette di controllare l'allocazione delle risorse in modo più granulare e flessibile.
- **VirtualFree**: È complementare a VirtualAlloc e viene utilizzata per liberare la memoria precedentemente allocata. Questa funzione è essenziale per la gestione della memoria dinamica, poiché garantisce che la memoria non più necessaria venga restituita al sistema, riducendo così il rischio di frammentazione e migliorando l'efficienza complessiva del sistema.



The screenshot shows the imports table for the process **Malware_U3_W2_L5.exe**. The table has columns for Module Name, Imports, OFTs, TimeStamp, ForwarderChain, Name RVA, and FTs (IAT). The **KERNEL32.dll** row is highlighted in blue, showing 44 imports. One of these imports is **szAnsi**, which is highlighted with a green border. The **szAnsi** row contains the following information:

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

Queste funzioni, tra le altre importate da **KERNEL32**, sono fondamentali per il funzionamento efficace e efficiente delle applicazioni Windows, fornendo gli strumenti necessari per la gestione delle risorse, la comunicazione inter-processo e l'allocazione dinamica della memoria.

Malware_U3_W2_L5

Sezione del malware

The screenshot shows a software interface with two main panes. The left pane is a tree view of the file structure for 'File: Malware_U3_W2_L5.exe', listing sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories, Section Headers, and Import Directory. The right pane is a table titled 'Malware_U3_W2_L5.exe' with columns for Name, Virtual Size, Virtual Address, Raw Size, Raw Address, Reloc Address, Linenumbers, Relocations N..., Linenumbers ..., and Characteristics. The table contains four rows corresponding to the sections listed in the tree view.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
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

.text

La sezione .text contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta avviato il software. Generalmente, questa è l'unica sezione di un file eseguibile che viene eseguita direttamente dalla CPU, mentre tutte le altre sezioni contengono dati o informazioni di supporto.

.rdata

La sezione .rdata include generalmente le informazioni sulle librerie e le funzioni importate ed esportate dall'eseguibile. Queste informazioni possono essere ricavate utilizzando strumenti come CFF Explorer.

.data

La sezione .data contiene tipicamente i dati e le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Una variabile è considerata globale quando non è definita all'interno del contesto di una funzione, ma è dichiarata globalmente ed è quindi accessibile da qualsiasi funzione all'interno dell'eseguibile.

Identificazione del Malware

Ricerca su VirusTotal

Effettuando una ricerca su VirusTotal tramite l'hash ricavato da CFF Explorer, è stato riscontrato che il malware in questione potrebbe essere un Trojan.

The screenshot shows the VirusTotal website interface. At the top, there is a search bar with the placeholder "URL, IP address, domain or file hash". Below the search bar, a message states: "We have changed our Privacy Notice and Terms of Use, effective July 18, 2024. You can view the updated [Privacy Notice](#) and [Terms of Use](#)". There are "Sign in" and "Sign up" buttons. A circular progress bar on the left indicates a "Community Score" of 38 out of 72. The main content area displays a file analysis for the hash `b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dc6efd3d8416a`. The file is identified as `Lab06-02.exe`. It has a size of 40.00 KB and was last modified 6 days ago. The file is categorized as an EXE. Below the file details, under "Popular threat labels", the label `trojan.r002c0pdm21` is circled in red. Other threat categories listed are `peexe`, `checks-network-adapters`, `runtime-modules`, `direct-cpu-clock-access`, and `armadillo`. The "Threat categories" section shows `trojan` as a selected category. The "Family labels" section shows `r002c0pdm21`.

4 - Ipotesi comportamento della funzionalità

Il codice in Assembly x86 implementa una funzione che verifica la connessione a Internet utilizzando la funzione “InternetGetConnectedState”. Prima di chiamare la funzione, viene creato lo stack e vengono inseriti i tre parametri che saranno passati alla funzione: “ecx”, “dwReserved”, e “lpdwFlags”. La funzione verifica lo stato della connessione e restituisce il valore nel registro “eax”, che viene poi salvato nella variabile locale “var_4”.

A questo punto viene effettuato un confronto tramite l'istruzione “cmp”, che esegue una sottrazione tra i due operandi. In questo caso confronta “var_4” e “0”. Se sono uguali, l'istruzione “jz” salta alla locazione “loc_40102B” e stampa a video un errore, indicando che non c'è connessione. Altrimenti, se “var_4” e “0” non sono uguali, il salto non verrà eseguito e il codice continuerà, stampando a video "Success: Internet Connection".

InternetGetConnectedState

Questa funzione verifica lo stato della connessione Internet su un sistema operativo Windows. Quando la funzione restituisce TRUE, la connessione Internet è attiva; se restituisce FALSE, non è presente alcuna connessione.

Dettagli della Funzione InternetGetConnectedState:

dwReserved

Nella funzione InternetGetConnectedState, il parametro dwReserved è riservato per usi futuri e attualmente non viene utilizzato.

lpdwFlags

Il prefisso "lp" indica un "long pointer", ovvero un puntatore a un valore DWORD. Nella funzione InternetGetConnectedState, il parametro lpdwFlags viene utilizzato per restituire lo stato della connessione Internet. La funzione scrive i dati relativi allo stato della connessione all'indirizzo di memoria puntato da lpdwFlags, consentendo al chiamante di ottenere tali informazioni dopo l'esecuzione della funzione.

5 - Costrutti noti e significato di tutte le righe del codice

push ebp	viene salvato il valore attuale di ebp, puntatore alla base dello stack, nel nuovo stack che stiamo creando. In questo modo il valore di ebp può essere ripristinato alla fine della funzione.
mov ebp, esp	Viene copiato il valore del registro esp, che è lo stack pointer, nel registro ebp. In questo modo il registro ebp punta alla funzione corrente e questo rende più facile l'accesso a variabili locali tramite offset rispetto a ebp.
push ecx	salva il valore attuale del registro ecx. Serve sempre per preservare il registro e ripristinarlo alla fine della chiamata.
push 0 ;dwReserved	Carica il valore 0 nello stack. Questo 0 rappresenta il parametro dwReserved che deve essere 0 perché è riservato e quindi non utilizzabile.
push 0 ;lpdwFlags	Carica un altro valore 0 nello stack. Questa volta lo 0 è associato al parametro lpdwFlags perché non si vogliono ulteriori informazioni riguardo il tipo di connessione.
call ds:InternetGetConnectedState	chiama la funzione tramite il registro segmentazione ds. La funzione "InternetGetConnectedState" è una funzione API di Windows che riceve i 3 parametri precedentemente caricati nello stack, e verifica se il sistema è connesso o meno ad internet.
mov [ebp+var_4], eax	Copia il valore del registro eax(utilizzato per contenere il valore di ritorno della funzione chiamata) nella variabile locale var_4 che si trova probabilmente ad un offset di -4 dal valore del registro ebp.
cmp [ebp+var_4], 0	confronta il valore appena salvato nella variabile var_4 (cioè eax) con lo 0
jz short loc_40102B	Jump if zero. Questa istruzione effettua un salto alla locazione loc_40102B se la sottrazione tra var_4 e 0 è uguale a 0, e quindi se i due valori sono uguali e di conseguenza il flag ZF = 1.
push offset aSuccessInterne ;"Success: Internet Connection\n"	carica nello stack l'indirizzo della stringa aSuccessInterne. Offset aSuccessInternet serve per ottenere l'indirizzo di memoria della stringa "Success: Internet Connection\n"
call sub_40117F	chiama la funzione sub_40117F che potrebbe essere progettata per stampare una stringa. In questo caso vien da sé che riceve come parametro l'indirizzo della stringa appena descritta sopra
add esp, 4	Aggiunge 4 al valore di esp. Questa istruzione serve per ripulire lo stack pointer rimuovendo, in questo caso, un argomento di 4 byte, cioè la variabile locale var_4
mov eax, 1	copia il valore 1 all'interno del registro eax
jmp short loc_40103A	salto incondizionato di breve distanza, "short"(-128 +127 byte), alla locazione loc_40103A

=costruzione dello stack

=costrutto if

loc_40102B:	
push offset aError1_1NoInte ; "Error 1.1: No Internet\n"	carica nello stack l'indirizzo della stringa aSuccessInternet. Offset aSuccessInternet serve per ottenere l'indirizzo di memoria della stringa "Success: Internet Connection\n"
call sub_40117F	chiama la funzione sub_40117F che potrebbe essere progettata per stampare una stringa. In questo caso vien da sé che riceve come parametro l'indirizzo della stringa appena descritta sopra
add esp, 4	Aggiunge 4 al valore di esp. Questa istruzione serve per ripulire lo stack pointer rimuovendo, in questo caso, un argomento di 4 byte, cioè la variabile locale var_4
xor eax, eax	inizializza a 0 il registro EAX. Infatti l'operatore logico tra due bit identici restituisce sempre 0.

loc_40103A:	
mov esp, ebp	copia il valore del registro ebp nel registro esp. Ripristina il valore dello stack pointer esp che aveva all'inizio della funzione eliminando tutte le variabili locali allocate sullo stack
pop ebp	rimuove dallo stack il registro ebp che era stato aggiunto all'inizio con push
retn	Restituisce il controllo alla funzione chiamante
sub_401000 ends	indica la fine della funzione. non è un'istruzione eseguibile.

=rimozione dello stack



TEAM ALBA



Zhongshi Liu



Mara Dello Russo



Mario Marsicano



Luca Lenzi



Giovanni Sannino



Andre Vinicius

THANK YOU
